Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 15 (2011) 3673 – 3677

Procedia Engineering

www.elsevier.com/locate/procedia

Advanced in Control Engineeringand Information Science

Application of Improved Discrete Particle Swarm Optimization in Logistics Distribution Routing Problem

Chengming Qi [a]*

Beijing Union University, Beijing 100101, China

Abstract

Particle swarm optimization (PSO) is an evolutionary metaheuristic inspired by the flocking behavior of birds, which has successfully been used to solve several kinds of problems. In this paper, we refer to capacitated vehicle routing problem (CVRP) as the object to research distribution routing optimistic problem within transportation and logistics. An improved discrete particle swarm optimization (DPSO), a variant of PSO, combined with iterated local search (ILS) method is proposed. During the search process, the improved algorithm can adaptively adjust flying time of particles with the evolutionary generations and ILS ensures that the particles escaped from the local minimum. The experimental results for partial benchmark problems showed that the better results can be obtained by the method.

Keywords: Capacitated Vehicle Routing Problem, Combinatorial optimization, Iterated Local Search, Particle swarm optimization;

1. Introduction

The vehicle routing problem (VRP) is well known combinatorial optimization problem with considerable economic significance. A typical VRP can be described as the problem of designing least cost routes from one depot to a set of geographically scattered points (cities, warehouses, customers etc).

[a] * Corresponding author. Tel.: 13520509271
E-mail address: qicm@163.com.

1877-7058 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.proeng.2011.08.688

The VRP has been largely studied because of the interest in its applications in logistic and supply-chains management.

The Particle Swarm Optimization (PSO) algorithm was originally introduced in [1] as a new, simple evolutionary algorithm, which differs from other evolution motivated evolutionary computation techniques in that it is motivated from the simulation of social behavior. The basic theory of the PSO algorithm is provided for the continuous problem. In 1997, Kennedy and others proposed a discrete binary PSO algorithm. And Clerec updated the formula changes in discrete particle swarm optimization (DPSO) algorithm problems for the first time. Recently, there have been a number of attempts to combine the use of DPSO algorithms with other techniques in order to locate multiple solutions of the problem find multiple optimal solutions. Afshinmanesh et al. [2] have developed a hybrid approach, combining PSO and Artificial Immune Systems (AIS), for the optimization of binary problems.

Local search is a generally applicable approach that can be used to find approximate solutions to hard optimization problems. The basic idea is to start from an initial solution and to search for successive improvements by examining neighboring solutions. In this paper, we apply iterated local search (ILS) [3] in second stage. ILS is a very simple and powerful metaheuristic that has proved to be the best performing approximation algorithms for the well known Traveling Salesman Problem [3].

In this paper, we propose an algorithm DPSOILS (Discrete Particle Swarm Optimization Hybridized with Iterated Local Search) to solve the problems of easily falling into local optimum solution. During the global searching process, the randomized local search operator can drive each particle to a neighbor local search area to enforce the local search ability of PSO.

2. Discrete Particle Swarm Optimization

2.1. Fundamental Principle of Particle Swarm Optimization

In PSO, each individual, named particle, of the population, called swarm, adjusts its trajectory toward its own previous best position, and toward the previous best position attained by any member of its topological neighborhood [4]. In the global variant of PSO, the whole swarm is considered as the neighborhood. Thus, global sharing of information takes place and the particles profit from the discoveries and previous experience of all other companions during the search for promising regions of the landscape.

We define the notation adopted in this paper: assuming that the search space is N-dimensional, the number of particle is *n*, the *i*-th particle of the swarm is represented by the *N*-dimensional vector $X_i=(x_{i1},x_{i2},\ldots,x_{iN})$ and the best particle of the swarm, i.e. the particle with the lowest function value, is denoted by index *g*. The best previous position (i.e. the position giving the best function value) of the *i*-th particle is recorded and represented by $X_i=(x_{i1},x_{i2},\ldots,x_{iN})$, and the position change (velocity) of the *i*-th particle is $V_i=(v_{i1},v_{i2},\ldots,v_{iN})$.

The particles are manipulated according to the following equations (the superscripts denote the iteration number):

$$v_{il}^{k+1}=\omega v_{il}^{k}+c_1\times rand_1()\times(p_{il}^{k}-x_{il}^{k})+c_2\times rand_2()\times(p_{gl}^{k}-x_{il}^{k}) \quad (1)$$

$$x_{id}=x_{id}+v_{id} \quad (2)$$

where $1\le i\le m$， $1\le d\le D$, w is the inertia weight, c_1 and c_2 are two positive constants, called the cognitive and social parameter respectively; $rand_1()$ and $rand_2()$ are two random numbers uniformly distributed within the range [0, 1]. Some variants of PSO impose a maximum allowed velocity V_{max} to prevent the swarm from explosion (i.e. if $v_{il}^{k}>v_{max}$, then $v_{il}^{k}=v_{max}$) [4].

Eq. (1) is used to calculate the *i*-th particle's new velocity, at each iteration. Three terms are taken into consideration. The first term, v_{il}^{k} , is the particle's previous velocity . The second term, p_{il}^{k}-x_{il}^{k}, is the distance between the particle's best previous position, and its current position. Finally, the third term, p_{gl}^{k}-x_{il}^{k}, is the distance between the swarm's best experience, and the *i*-th particle's current position. The parameters *c1*, *c2, rand*$_1$ *()* and *rand*$_2$ *()*, provide randomness that makes the technique less predictable yet more flexible [4].

Eq. (2) provides the new position of the *i*-th particle, adding its new velocity, to its current position. The inertia weight is employed to control the impact of the previous history of velocities on the current velocity. In this way, the parameter *w* regulates the trade-off between the global and local exploration abilities of the swarm and influences PSO convergence behaviour. A large inertia weight facilitates global exploration (searching new areas) while a small one tends to facilitate local exploration

2.2. Discrete Particle Swarm Optimization

Although the main body of PSO development work has concentrated on optimization in continuous search spaces, research has also been conducted into the application of the algorithm to discrete problems. Kennedy and Eberhart [5] developed the first discrete version of PSO for binary problems. They solved the problem of moving the particle through the problem space by changing the velocity in each vector to the probability of each bit being in one state or the other.

Each swarm contains *M* particles with randomly initialized positions and velocities in an *n*-dimensional search space. Each particle stores its current position *x*, current velocity *v*, and the best position it has visited so far *l*, and (in the most frequently used version) is also presumed to know the best *g* among all positions visited so far by all particles in the swarm. The particle moves according to the dynamics described below.

- The velocity is incremented by a weighted sum of *l*-*x* and *g*-*x*; the magnitude of each velocity component is often limited.
- The position is incremented by the velocity. Particles move until termination criteria are met, such as a user-defined maximum number of iterations (discrete time steps) or a satisfactory solution is found [1].

For binary discrete search spaces, Kennedy and Eberhart [5] use the above velocity update equation but compute the actual new position component to be 1 with a probability obtained by applying a sigmoid transformation $1/(1+e^{-v})$ to the velocity component.

3. Discrete Particle Swarm Optimization for the CVRP

In this section, we will introduce CVRP and apply ant colony system to the CVRP.

3.1. Capacitated Vehicle Routing Problem

The Capacitated Vehicle Routing Problem (CVRP) is the basic version of the VRP. The name derives from the constraint of having vehicles with limited capacity. The CVRP is NP-hard [6], since it contains one or more TSP as subproblems. Obviously, a CVRP is more difficult to solve than a TSP.

The classic CVRP can be described as follows: *N* customers geographically dispersed in a planar region must be served from a unique depot. Each customer asks for a quantity q_i *(i=1,2,...,n)* of goods. The transport cost from node *i* to node *j* is c_{ij}. *m* vehicles with a fixed capacity *Q* are available to deliver the goods stored in the depot. Each customer must be visited just once by only one vehicle. The objective of the problem is minimizing the total cost of all routes without violating the individual capacity of each vehicle. The depot is denoted by *i=0*.

3.2. DPSO hybridized with ILS

In this subsection, we propose an improved algorithm DPSOILS. The approach applies iterated local search algorithm to PSO.

ILS is a very simple and powerful stochastic local search method that has proved to be among the best performing approximation algorithms for the well-known Traveling Salesman Problem (TSP) [7] and a number of other problems [8]. The essential idea of ILS is to perform a biased, randomized walk in the space of locally optimal solutions instead of sampling the space of all possible candidate solutions.

To apply an ILS algorithm, basically three procedures have to be specified. Given the current s_0, these are a procedure *Perturbation*, that perturbs the current solution s leading to same intermediate solution s', a procedure *LocalSearch* that takes s' to a local optimum s'', and an *AcceptanceCriterion* that decides from which solution the next perturbation step is applied [8].

Each solution established in the former stage s_0 is taken to a local optimum. Then, iteratively a local search procedure is applied from a starting solution obtained by a perturbation of the current search point. In this article, the main idea behind the DPSOILS extension is to use ILS to add diversity. The only difference between a standard particle and a DPSOILS particle is an additional random perturbation for every element of velocity vector v_i as description in Eq.(2'). Random perturbation is added according to the difference of particles in different position, which can balance local and globe search scope.

$$x_{id}=x_{id}+rand()(1-k\times iter/I_{max})\times v_{id} \qquad (2')$$

Where rand() is random number uniformly distributed within the range [0, 1]. I_{max} is the total number of iterations. iter is current iteration size. Constant k balances I_{max} with iter. Usually, there is $I_{max}>>$iter.

The PSOILS model is similar to the traditional PSO model. The model consists of a number of particles moving around in the search space, where the position of each particle represents a candidate solution to a numerical problem. Each particle has a position vector x_i, a velocity vector v_i. The velocity v_i of each particle is limited by an upper threshold v_{max}. The position of each particle is updated in each iteration by adding the velocity vector obtained from iterated local search to the position vector. The particles have no neighborhood restrictions, meaning that each particle can affect all other particles.

4. Experimental results

We now evaluate the performance of DPSOILS on 12 instances of CVRP benchmark problems. These include the best-known solutions to each problem. These problems range from 32 customers to 80 customers and from 5 vehicles to 14 vehicles for the solution. For each instance of the datasets, the number of customers is given by the first number on the instance name. The main difference between these sets of problems is their tightness (the ratio between demand and capacity) and the location of customers.

Experiments were run on a Pentium IV, 1GB of RAM, 2.0 GHz processor. Solutions are then averaged for each problem type and the result is reported in Table 1. We used n=20 artificial ants and set α= 1, q_0 = 0.9, β = 2 and ρ= 0.1. For all problems maximum iteration times are $2*n=40$.

In Table 1, we present the results achieved by DPSOILS. The table shows the best solutions found by the proposed algorithm as well as the averages of the best solution found in each of the 50 runs. The column *Optimum* indicates the best known solution when our research started. The results reveal that DPSOILS was able to find the best solutions for all instances. The averages obtained with DPSOILS are close to the values of the best solutions. In fact, the distances can be lower than 3%. Another interesting point is that DPSOILS was able to find so far the best solutions (instances A32k5, E76k7, E76k8, and E76k14).

Table 1. Results from DPSOILS

Instances	Optimum	*DPSOILSBest*	*DPSOILSAvg*
A32k5	784	784	792.1
A54k7	1167	1170	1180.5
A60k9	1358	1364	1375.6
A69k9	1167	1169	1181.7
A80k10	1764	1775	1813.9
B57k7	1153	1163	1176.2
B63k10	1496	1532	1534.2
B78k10	1266	1272	1275.6
E76k7	682	682	686.6
E76k8	735	735	738.3
E76k10	832	846	861.3
E76k14	1032	1032	1040.5

5. Conclusion

In this paper, an improved discrete particle swarm optimizer with iterated local search was introduced to improve the performance. Computational results shows the performance of the DPSOILS approach can generate very high quality solutions for the CVRP, and proved that DPSOILS is an interesting novel stochastic approach to optimization of the VRP.

As future work, we intend to perform a detailed study on the importance of DPSOILS, to conduct experiments to evaluate the effectiveness of DPSOILS on real-world problems and wider range of combinatorial problems.

References

[1] Kennedy J, Eberhart R C. Particle Swarm Optimization [A]. Proc of the IEEE Intl. Conf. on Neural Networks[C], pages 1942-1948. 1995.

[2] Afshinmanesh F, Marandi A, Rahimi-Kian A (2005) A novel binary particle swarm optimization method using artificial immune system. EuroCon 2005-The international conference on computer as a tool,Serbia & Montenegro, Belgrade.

[3] O. Martin, S.W. Otto, and E.W. Felten. Large-Step Markov Chains for the Traveling Salesman Problem. Complex Systems, 1991,5(3),pp. 299-326.

[4] Kennedy J. The Behavior of Particles. Evolutionary Programming VII, pages 581-587, 1998.

[5] Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. Proceedings of the conference on systems, man and cybernetics, Piscataway, New Jersey, pp 4104-4109

[6] Lenstr, J.K. & Rinnooy Kan, A.H.G., 1981. Complexity of vehicle routing and scheduling problems. Networks, vol. 11, pp. 221-227.

[7] D.S. Johnson, L.A. McGeoch, Experimental analysis of heuristics for the STSP, in: G. Gutin, A. Punnen (Eds.), The Traveling Salesman Problem and Its Variations, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002, pp. 369-443.

[8] T. Stützle. Applying iterated local search to the permutation flow shop problem. Technical Report AIDA-98-04, FG Intellektik, TU Darmstadt, August 1998.